THE COMPLEXITY OF THE LIST HOMOMORPHISM PROBLEM FOR GRAPHS

LÁSZLÓ EGRI [1] AND ANDREI KROKHIN [2] AND BENOIT LAROSE [3] AND PASCAL TESSON [4]

[1] School of Computer Science, McGill University, Montréal, Canada
  E-mail address: `laszlo.egri@mail.mcgill.ca`

[2] School of Engineering and Computing Sciences, Durham University, Durham, UK
  E-mail address: `andrei.krokhin@durham.ac.uk`

[3] Department of Mathematics and Statistics, Concordia University, Montréal, Canada
  E-mail address: `larose@mathstat.concordia.ca`

[4] Department of Computer Science, Laval University, Quebec City, Canada
  E-mail address: `pascal.tesson@ift.ulaval.ca`

ABSTRACT. We completely classify the computational complexity of the list **H**-colouring problem for graphs (with possible loops) in combinatorial and algebraic terms: for every graph **H** the problem is either NP-complete, NL-complete, L-complete or is first-order definable; descriptive complexity equivalents are given as well via Datalog and its fragments. Our algebraic characterisations match important conjectures in the study of constraint satisfaction problems.

1. Introduction

Homomorphisms of graphs, i.e. edge-preserving mappings, generalise graph colourings, and can model a wide variety of combinatorial problems dealing with mappings and assignments [17]. Because of the richness of the homomorphism framework, many computational aspects of graph homomorphisms have recently become the focus of much attention. In the *list* **H**-*colouring* problem (for a fixed graph **H**), one is given a graph **G** and a list L_v of vertices of **H** for each vertex v in **G**, and the goal is to determine whether there is a homomorphism h from **G** to **H** such that $h(v) \in L_v$ for all v. The complexity of such problems has been studied by combinatorial methods, e.g., in [13, 14]. In this paper, we study the complexity of the list homomorphism problem for graphs in the wider context of classifying the complexity of constraint satisfaction problems (CSP), see [3, 15, 18]. It is well known that the CSP can be viewed as the problem of deciding whether there exists a homomorphism from a relational structure to another, thus naturally extending the graph homomorphism problem.

Key words and phrases: graph homomorphism, constraint satisfaction problem, complexity, universal algebra, Datalog.

One line of CSP research studies the non-uniform CSP, in which the target (or template) structure $\mathbf{T}$ is fixed and the question is whether there exists a homomorphism from an input structure to $\mathbf{T}$. Over the last years, much work has been done on classifying the complexity of this problem, denoted $\mathrm{Hom}(\mathbf{T})$ or $\mathrm{CSP}(\mathbf{T})$, with respect to the fixed target structure, see surveys [6, 7, 8, 18]. Classification here is understood with respect to both computational complexity (i.e. membership in a given complexity class such as P, NL, or L, modulo standard assumptions) and descriptive complexity (i.e. definability of the class of all positive, or all negative, instances in a given logic).

The best-known classification results in this direction concern the distinction between polynomial-time solvable and NP-complete CSPs. For example, a classical result of Hell and Nešetřil (see [17, 18]) shows that, for a graph $\mathbf{H}$, $\mathrm{Hom}(\mathbf{H})$ (aka $\mathbf{H}$-colouring) is tractable if $\mathbf{H}$ is bipartite or admits a loop, and is NP-complete otherwise, while Schaefer's dichotomy [24] proves that any Boolean CSP is either in P or NP-complete. Recent work [1] established a more precise classification in the Boolean case: if $\mathbf{T}$ is a structure on $\{0, 1\}$ then $\mathrm{CSP}(\mathbf{T})$ is either NP-complete, P-complete, NL-complete, $\oplus$L-complete, L-complete or in AC^0.

Much of the work concerning the descriptive complexity of CSPs is centred around the database-inspired logic programming language Datalog and its fragments (see [6, 9, 12, 15, 20]). Feder and Vardi initially showed [15] that a number of important tractable cases of $\mathrm{CSP}(\mathbf{T})$ correspond to structures for which $\neg\,\mathrm{CSP}(\mathbf{T})$ (the complement of $\mathrm{CSP}(\mathbf{T})$) is definable in Datalog. Similar ties were uncovered more recently between the two fragments of Datalog known as linear and symmetric Datalog and structures $\mathbf{T}$ for which $\mathrm{CSP}(\mathbf{T})$ belongs to NL and L, respectively [9, 12].

Algebra, logic and combinatorics provide three angles of attack which have fueled progress in this classification effort [6, 7, 8, 17, 18, 20]. The algebraic approach (see [7, 8]) links the complexity of $\mathrm{CSP}(\mathbf{T})$ to the set of functions that preserve the relations in $\mathbf{T}$. In this framework, one associates to each $\mathbf{T}$ an algebra $\mathbb{A}_{\mathbf{T}}$ and exploits the fact that the properties of $\mathbb{A}_{\mathbf{T}}$ completely determine the complexity of $\mathrm{CSP}(\mathbf{T})$. This angle of attack was crucial in establishing key results in the field (see, for example, [2, 5, 7]).

Tame Congruence Theory, a deep universal-algebraic framework first developed by Hobby and McKenzie in the mid 80's [19], classifies the local behaviour of finite algebras into five *types* (unary, affine, Boolean, lattice and semilattice.) It was recently shown (see [6, 7, 22]) that there is a strong connection between the computational and descriptive complexity of $\mathrm{CSP}(\mathbf{T})$ and the set of types that appear in $\mathbb{A}_{\mathbf{T}}$ and its subalgebras. There are strong conditions involving types which are sufficient for NL-hardness, P-hardness and NP-hardness of $\mathrm{CSP}(\mathbf{T})$ as well as for inexpressibility of $\neg\,\mathrm{CSP}(\mathbf{T})$ in Datalog, linear Datalog and symmetric Datalog. These sufficient conditions are also suspected (and in some cases proved) to be necessary, under natural complexity-theoretic assumptions. For example, (a) the presence of unary type is known to imply NP-completeness, while its absence is conjectured to imply tractability (see [7]); (b) the absence of unary and affine types was recently proved to be equivalent to definability in Datalog [2]; (c) the absence of unary, affine, and semilattice types is proved necessary, and suspected to be sufficient, for membership in NL and definability in linear Datalog [22]; (d) the absence of all types but Boolean is proved necessary, and suspected to be sufficient, for membership in L and definability in symmetric Datalog [22]. The strength of evidence varies from case to case and, in particular, the conjectured algebraic conditions concerning CSPs in NL and L (and, as mentioned above, linear and symmetric Datalog) still rest on relatively limited evidence [6, 9, 11, 10, 22].

The aim of the present paper is to show that these algebraic conditions are indeed sufficient and necessary in the special case of list **H**-colouring for undirected graphs (with possible loops), and to characterise, in this special case, the dividing lines in graph-theoretic terms (both via forbidden subgraphs and through an inductive definition). One can view the list **H**-colouring problem as a CSP where the template is the structure $\mathbf{H}^L$ consisting of the binary (edge) relation of **H** and all unary relations on H (i.e. every subset of H). Tractable list homomorphism problems for general structures were characterised in [5] in algebraic terms. The tractable cases for graphs were described in [14] in both combinatorial and (more specific) algebraic terms; the latter implies, when combined with a recent result [10], that in these cases $\neg\,\mathrm{CSP}(\mathbf{H}^L)$ definable in linear Datalog and therefore $\mathrm{CSP}(\mathbf{H}^L)$ is in fact in NL. We complete the picture by refining this classification and showing that $\mathrm{CSP}(\mathbf{H}^L)$ is either NP-complete, or NL-complete, or L-complete or in AC^0 (and in fact first-order definable). We also remark that the problem of recognising into which case the problem $\mathrm{CSP}(\mathbf{H}^L)$ falls can be solved in polynomial time.

As we mentioned above, the distinction between NP-complete cases and those in NL follows from earlier work [14], and the situation is similar with distinction between L-hard cases and those leading to membership in AC^0 [21, 22]. Therefore, the main body of technical work in the paper concerns the distinction between NL-hardness and membership in L. We give two equivalent characterisations of the class of graphs **H** such that $\mathrm{CSP}(\mathbf{H}^L)$ is in L. One characterisation is via forbidden subgraphs (for example, the reflexive graphs in this class are exactly the (P_4, C_4)-free graphs, while the irreflexive ones are exactly the bipartite (P_6, C_6)-free graphs), while the other is via an inductive definition. The first characterisation is used to show that graphs outside of this class give rise to NL-hard problems; we do this by providing constructions witnessing the presence of a non-Boolean type in the algebras associated with the graphs. The second characterisation is used to prove positive results. We first provide operations in the associated algebra which satisfy certain identities; this allows us to show that the necessary condition on types is also sufficient in our case. We also use the inductive definition to demonstrate that the class of negative instances of the corresponding CSP is definable in symmetric Datalog, which implies membership of the CSP in L.

2. Preliminaries

2.1. Graphs and relational structures

In the following we denote the underlying universe of a structure **S**, **T**, ... by its roman equivalent S, T, etc. A signature is a (finite) set of relation symbols with associated arities. Let **T** be a structure of signature τ; for each relation symbol $R \in \tau$ we denote the corresponding relation of **T** by $R(\mathbf{T})$. Let **S** be a structure of the same signature. A *homomorphism* from **S** to **T** is a map f from S to T such that $f(R(\mathbf{S})) \subseteq R(\mathbf{T})$ for each $R \in \tau$. In this case we write $f : \mathbf{S} \to \mathbf{T}$. A structure **T** is called a *core* if every homomorphism from **T** to itself is a permutation on T. We denote by $\mathrm{CSP}(\mathbf{T})$ the class of all τ-structures **S** that admit a homomorphism to **T**, and by $\neg\,\mathrm{CSP}(\mathbf{T})$ the complement of this class.

The *direct n-th power* of a τ-structure **T**, denoted $\mathbf{T}^n$, is defined to have universe T^n and, for any (say m-ary) $R \in \tau$, $(\mathbf{a}_1, \ldots, \mathbf{a}_m) \in R(\mathbf{T}^n)$ if and only if $(\mathbf{a}_1[i], \ldots, \mathbf{a}_m[i]) \in R(\mathbf{T})$ for each $1 \le i \le n$. For a subset $I \subseteq T$, the *substructure induced by I on* **T** is the structure **I** with universe I and such that $R(\mathbf{I}) = R(\mathbf{T}) \cap I^m$ for every m-ary $R \in \tau$.

For the purposes of this paper, a *graph* is a relational structure $\mathbf{H} = \langle H; \theta \rangle$ where θ is a symmetric binary relation on H. The graph $\mathbf{H}$ is *reflexive (irreflexive)* if $(x,x) \in \theta$ ($(x,x) \notin \theta$) for all $x \in H$. Given a graph $\mathbf{H}$, let $S_1, \ldots, S_k$ denote all subsets of H; let $\mathbf{H}^L$ be the relational structure obtained from $\mathbf{H}$ by adding all the S_i as unary relations; more precisely, let τ be the signature that consists of one binary relational symbol θ and unary symbols R_i, $i = 1, \ldots, k$. The τ-structure $\mathbf{H}^L$ has universe H, $\theta(\mathbf{H}^L)$ is the edge relation of $\mathbf{H}$, and $R_i(\mathbf{H}^L) = S_i$ for all $i = 1, \ldots, k$. It is easy to see that $\mathbf{H}^L$ is a core. We call CSP($\mathbf{H}^L$) the *list homomorphism problem for* $\mathbf{H}$. Note that if $\mathbf{G}$ is an instance of this problem then $\theta(\mathbf{G})$ can be considered as a digraph, but the directions of the arcs are unimportant because $\mathbf{H}$ is undirected. Also, if an element $v \in G$ is in $R_i(\mathbf{G})$ then this is equivalent to v having S_i as its list, so $\mathbf{G}$ can be thought of as a digraph with $\mathbf{H}$-lists.

In [14], a dichotomy result was proved, identifying bi-arc graphs as those whose list homomorphism problem is tractable, and others as giving rise to NP-complete problems. Let C be a circle with two specified points p and q. A bi-arc is a pair of arcs (N, S) such that N contains p but not q and S contains q but not p. A graph $\mathbf{H}$ is a *bi-arc graph* if there is a family of bi-arcs $\{(N_x, S_x) : x \in H\}$ such that, for every $x, y \in H$, the following hold: (i) if x and y are adjacent, then neither N_x intersects S_y nor N_y intersects S_x, and (ii) if x is not adjacent to y then both N_x intersects S_y and N_y intersects S_x.

2.2. Algebra

An *n*-ary operation on a set A is a map $f : A^n \to A$, a *projection* is an operation of the form $e_n^i(x_1, \ldots, x_n) = x_i$ for some $1 \leq i \leq n$. Given an h-ary relation θ and an n-ary operation f on the same set A, we say that f *preserves* θ or that θ is *invariant* under f if the following holds: given any matrix M of size $h \times n$ whose columns are in θ, applying f to the rows of M will produce an h-tuple in θ.

A *polymorphism* of a structure $\mathbf{T}$ is an operation f that preserves each relation in $\mathbf{T}$; in this case we also say that $\mathbf{T}$ *admits* f. In other words, an n-ary polymorphism of $\mathbf{T}$ is simply a homomorphism from $\mathbf{T}^n$ to $\mathbf{T}$. With any structure $\mathbf{T}$, one associates an algebra $\mathbb{A}_{\mathbf{T}}$ whose universe is T and whose operations are all polymorphisms of $\mathbf{T}$. Given a graph $\mathbf{H}$, we let $\mathbb{H}$ denote the algebra associated with $\mathbf{H}^L$. An operation on a set is called *conservative* if it preserves all subsets of the set (as unary relations). So, the operations of $\mathbb{H}$ are the conservative polymorphisms of $\mathbf{H}$. Polymorphisms can provide a convenient language when defining classes of graphs. For example, it was shown in [4] that a graph is a bi-arc graph if and only if it admits a conservative majority operation where a *majority* operation is a ternary operation m satisfying the identities $m(x, x, y) = m(x, y, x) = m(y, x, x) = x$.

In order to state some of our results, we will need the notions of a variety and a term operation. Let I be a signature, i.e. a set of operation symbols f each of a fixed arity (we use the term "signature" for both structures and algebras, this will cause no confusion). An *algebra* of signature I is a pair $\mathbb{A} = \langle A; F \rangle$ where A is a non-empty set (the *universe of* $\mathbb{A}$) and $F = \{f^{\mathbb{A}} : f \in I\}$ is the set of *basic* operations (for each $f \in I$, $f^{\mathbb{A}}$ is an operation on A of the corresponding arity). The *term operations* of $\mathbb{A}$ are the operations built from the operations in F and projections by using composition. An algebra all of whose (basic or term) operations are conservative is called a *conservative algebra*. A class of similar algebras (i.e. algebras with the same signature) which is closed under formation of homomorphic images, subalgebras and direct products is called a *variety*. The *variety generated* by an

algebra $\mathbb{A}$ is denoted by $\mathcal{V}(\mathbb{A})$, and is the smallest variety containing $\mathbb{A}$, i.e. the class of all homomorphic images of subalgebras of powers of $\mathbb{A}$.

Tame Congruence Theory, as developed in [19], is a powerful tool for the analysis of finite algebras. Every finite algebra has a *typeset*, which describes (in a certain specified sense) the local behaviour of the algebra. It contains one or more of the following 5 *types*: (1) the *unary* type, (2) the *affine* type, (3) the *Boolean* type, (4) the *lattice* type and (5) the *semilattice* type. The numbering of the types is fixed, and they are often referred to by their numbers. Simple algebras, i.e. algebras without non-trivial proper homomorphic images, admit a unique type; the prototypical examples are: any 2-element algebra whose basic operations are all unary has type 1. A finite vector space has type 2. The 2-element Boolean algebra has type 3. The 2-element lattice is the 2-element algebra with two binary operations $\langle\{0,1\};\vee,\wedge\rangle$: it has type 4. The 2-element semilattices are the 2-element algebras with a single binary operation $\langle\{0,1\};\wedge\rangle$ and $\langle\{0,1\};\vee\rangle$: they have type 5. The typeset of a variety $\mathcal{V}$, denoted $typ(\mathcal{V})$, is simply the union of typesets of the algebras in it. We will be mostly interested in type-omitting conditions for varieties of the form $\mathcal{V}(\mathbb{A}_{\mathbf{T}})$, and Corollary 3.2 of [25] says that in this case it is enough to consider the typesets of $\mathbb{A}_{\mathbf{T}}$ and its subalgebras.

On the intuitive level, if $\mathbf{T}$ is a core structure then the typeset $typ(\mathcal{V}(\mathbb{A}_{\mathbf{T}}))$ contains crucial information about the kind of relations that $\mathbf{T}$ can or cannot simulate, thus implying lower/upper bounds on the complexity of $\mathrm{CSP}(\mathbf{T})$. For our purposes here, it will not be necessary to delve further into the technical aspects of types and typesets. We only note that there is a very tight connection between the kind of equations that are satisfied by the algebras in a variety and the types that are *admitted* or *omitted* by a variety, i.e. those types that do or do not appear in the typesets of algebras in the variety [19].

In this paper, we use ternary operations $f_1,\ldots,f_n$ satisfying the following identities:

$$x \quad = \quad f_1(x,y,y) \tag{2.1}$$

$$f_i(x,x,y) \quad = \quad f_{i+1}(x,y,y) \text{ for all } i=1,\ldots n-1 \tag{2.2}$$

$$f_n(x,x,y) \quad = \quad y. \tag{2.3}$$

The following lemma contains some type-omitting results that we use in this paper.

Lemma 2.1. [19] *A finite algebra $\mathbb{A}$ has term operations $f_1,\ldots,f_n$, for some $n \geq 1$, satisfying identities (2.1)–(2.3) if and only if the variety $\mathcal{V}(\mathbb{A})$ omits types 1, 4 and 5. If a finite algebra $\mathbb{A}$ has a majority term operation then $\mathcal{V}(\mathbb{A})$ omits types 1, 2 and 5.*

We remark in passing that operations satisfying identities (2.1)–(2.3) are also known to characterise a certain algebraic (congruence) condition called $(n+1)$-permutability [19].

2.3. Datalog

Datalog is a query and rule language for deductive databases (see [20]). A Datalog program $\mathcal{D}$ over a (relational) signature τ is a finite set of rules of the form $h \leftarrow b_1 \wedge \ldots \wedge b_m$ where h and each b_i are atomic formulas $R_j(v_1,...,v_k)$. We say that h is the *head* of the rule and that $b_1 \wedge \ldots \wedge b_m$ is its *body*. Relational predicates R_j which appear in the head of some rule of $\mathcal{D}$ are called *intensional database predicates* (*IDB*s) and are not part of the signature τ. All other relational predicates are called *extensional database predicates* (*EDB*s) and are in τ. So, a Datalog program is a recursive specification of IDBs (from EDBs).

A rule of $\mathcal{D}$ is *linear* if its body contains at most one IDB and is *non-recursive* if its body contains only EDBs. A linear but recursive rule is of the form $I_1(\bar{x}) \leftarrow I_2(\bar{y}) \wedge E_1(\bar{z}_1) \wedge \ldots \wedge E_k(\bar{z}_k)$ where I_1, I_2 are IDBs and the E_i are EDBs (note that the variables occurring in $\bar{x}, \bar{y}, \bar{z}_i$ are not necessarily distinct). Each such rule has a *symmetric* $I_2(\bar{y}) \leftarrow I_1(\bar{x}) \wedge E_1(\bar{z}_1) \wedge \ldots \wedge E_k(\bar{z}_k)$. A Datalog program is *non-recursive* if all its rules are non-recursive, *linear* if all its rules are linear and *symmetric* if it is linear and if the symmetric of each recursive rule of $\mathcal{D}$ is also a rule of $\mathcal{D}$.

A Datalog program $\mathcal{D}$ takes a τ-structure $\mathbf{A}$ as input and returns a structure $\mathcal{D}(\mathbf{A})$ over the signature $\tau' = \tau \cup \{I : I \text{ is an IDB in } \mathcal{D}\}$. The relations corresponding to τ are the same as in $\mathbf{A}$, while the new relations are recursively computed by $\mathcal{D}$, with semantics naturally obtained via least fixed-point of monotone operators. We also want to view a Datalog program as being able to accept or reject an input τ-structure and this is achieved by choosing one of the IDBs of $\mathcal{D}$ as the *goal predicate*: the τ-structure $\mathbf{A}$ is *accepted by* $\mathcal{D}$ if the goal predicate is non-empty in $\mathcal{D}(\mathbf{A})$. Thus every Datalog program with a goal predicate defines a class of structures - those that are accepted by the program.

When using Datalog to study $\mathrm{CSP}(\mathbf{T})$, one usually speaks of the definability of $\neg\,\mathrm{CSP}(\mathbf{T})$ in Datalog (i.e. by a Datalog program) or its fragments (because any class definable in Datalog must be closed under extension). Examples of CSPs definable in Datalog and its fragments can be found, e.g., in [6, 12]. As we mentioned before, any problem $\mathrm{CSP}(\mathbf{T})$ is tractable if its complement is definable in Datalog, and all such structures were recently identified in [2]. Definability of $\neg\,\mathrm{CSP}(\mathbf{T})$ in linear (symmetric) Datalog implies that $\mathrm{CSP}(\mathbf{T})$ belongs to NL and L, respectively [9, 12]. As we discussed in Section 1, there is a connection between definability of CSPs in Datalog (and its fragments) and the presence/absence of types in the corresponding algebra (or variety).

Note that it follows from Lemma 2.1 and from the results in [22, 26] that if, for a core structure $\mathbf{T}$, $\neg\,\mathrm{CSP}(\mathbf{T})$ is definable in symmetric Datalog then $\mathbf{T}$ must admit, for some n, operations satisfying identities (2.1)–(2.3). Moreover, with the result of [2], a conjecture from [22] can be restated as follows: for a core structure $\mathbf{T}$, if $\neg\,\mathrm{CSP}(\mathbf{T})$ is definable in Datalog and, for some n, $\mathbf{T}$ admits operations satisfying (2.1)–(2.3), then $\neg\,\mathrm{CSP}(\mathbf{T})$ is definable in symmetric Datalog. This conjecture is proved in [11] for $n = 1$.

3. A class of graphs

In this section, we give combinatorial characterisations of a class of graphs whose list homomorphism problem will turn out to belong to L.

Let $\mathbf{H}_1$ and $\mathbf{H}_2$ be bipartite irreflexive graphs, with colour classes B_1, T_1 and B_2 and T_2 respectively, with T_1 and B_2 non-empty. We define the *special sum* $\mathbf{H}_1 \odot \mathbf{H}_2$ (which depends on the choice of the B_i and T_i) as follows: it is the graph obtained from the disjoint union of $\mathbf{H}_1$ and $\mathbf{H}_2$ by adding all possible edges between the vertices in T_1 and B_2. Notice that we can often decompose a bipartite graph in several ways, and even choose B_1 or T_2 to be empty. We say that an irreflexive graph $\mathbf{H}$ is *a special sum* or *expressed as a special sum* if there exist two bipartite graphs and a choice of colour classes on each such that $\mathbf{H}$ is isomorphic to the special sum of these two graphs.

Definition 3.1. Let $\mathcal{K}$ denote the smallest class of irreflexive graphs containing the one-element graph and closed under (i) special sum and (ii) disjoint union. We call the graphs in $\mathcal{K}$ *basic irreflexive*.
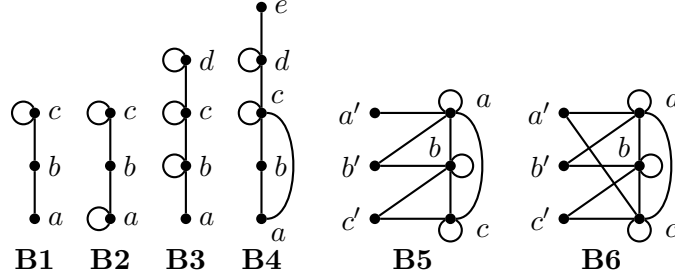
Figure 1: The forbidden mixed graphs.

The following result gives a characterisation of basic irreflexive graphs in terms of forbidden subgraphs:

Lemma 3.2. *Let* **H** *be an irreflexive graph. Then the following conditions are equivalent:*
  (1) **H** *is basic irreflexive;*
  (2) **H** *is bipartite, contains no induced 6-cycle, nor any induced path of length 5.*

We shall now describe our main family of graphs, first by forbidden induced subgraphs, and then in an inductive manner.

Definition 3.3. Define the class $\mathcal{L}$ of graphs as follows: a graph **H** belongs to $\mathcal{L}$ if it contains none of the following as an induced subgraph:
  (1) the reflexive path of length 3 and the reflexive 4-cycle;
  (2) the irreflexive cycles of length 3, 5 and 6, and the irreflexive path of length 5;
  (3) **B1**, **B2**, **B3**, **B4**, **B5** and **B6** (see Figure 1.)

We will now characterise the class $\mathcal{L}$ in an inductive manner.

Definition 3.4. A connected graph **H** is *basic* if either (i) **H** is a single loop, or (ii) **H** is a basic irreflexive graph, or (iii) **H** is obtained from a basic irreflexive graph $\mathbf{H}_1$ with colour classes B and T by adding every edge (including loops) of the form $\{t, t'\}$ where $t, t' \in T$.

Definition 3.5. Given two vertex-disjoint graphs $\mathbf{H}_1$ and $\mathbf{H}_2$, the *adjunction of* $\mathbf{H}_1$ *to* $\mathbf{H}_2$ is the graph $\mathbf{H}_1 \oslash \mathbf{H}_2$ obtained by taking the disjoint union of the two graphs, and adding every edge of the form $\{x, y\}$ where x is a loop in $\mathbf{H}_1$ and y is a vertex of $\mathbf{H}_2$.

Lemma 3.6. *Let* $\mathcal{L}_R$ *denote the class of reflexive graphs in* $\mathcal{L}$*. Then* $\mathcal{L}_R$ *is the smallest class* $\mathcal{D}$ *of reflexive graphs such that:*
  (1) $\mathcal{D}$ *contains the one-element graph;*
  (2) $\mathcal{D}$ *is closed under disjoint union;*
  (3) *if* $\mathbf{H}_1$ *is a single loop and* $\mathbf{H}_2 \in \mathcal{D}$ *then* $\mathbf{H}_1 \oslash \mathbf{H}_2 \in \mathcal{D}$*.*

Lemma 3.6 states that the reflexive graphs avoiding the path of length 3 and the 4-cycle are precisely those constructed from the one-element loop using disjoint union and adjunction of a universal vertex. These graphs can also be described by the following property: every connected induced subgraph of size at most 4 has a universal vertex. These graphs have been studied previously as those with NLCT width 1, which were proved to be exactly the trivially perfect graphs [16]. Our result provides an alternative proof of the equivalence of these conditions.

Theorem 3.7. *The class* $\mathcal{L}$ *is the smallest class* $\mathcal{C}$ *of graphs such that:*

(1) $\mathcal{C}$ *contains the basic graphs;*
(2) $\mathcal{C}$ *is closed under disjoint union;*
(3) *if* $\mathbf{H}_1$ *is a basic graph and* $\mathbf{H}_2 \in \mathcal{C}$ *then* $\mathbf{H}_1 \oslash \mathbf{H}_2 \in \mathcal{C}$.

Proof. We start by showing that every basic graph is in $\mathcal{L}$, i.e. that a basic graph does not contain any of the forbidden graphs. If $\mathbf{H}$ is a single loop or a basic irreflexive graph, then this is immediate. Otherwise $\mathbf{H}$ is obtained from a basic irreflexive graph $\mathbf{H}_1$ with colour classes B and T by adding every edge of the form (t_1, t_2) where $t_i \in T$. In particular, the loops form a clique and no edge connects two non-loops; it is clear in that case that $\mathbf{H}$ contains none of $\mathbf{B1}$, $\mathbf{B2}$, $\mathbf{B3}$, $\mathbf{B4}$. On the other hand if $\mathbf{H}$ contains $\mathbf{B5}$ or $\mathbf{B6}$, then $\mathbf{H}_1$ contains the path of length 5 or the 6-cycle, contradicting the fact that $\mathbf{H}_1$ is basic.

Next we show that $\mathcal{L}$ is closed under disjoint union and adjunction of basic graphs. It is obvious that the disjoint union of graphs that avoid the forbidden graphs will also avoid these. So suppose that an adjunction $\mathbf{H}_1 \oslash \mathbf{H}_2$, where $\mathbf{H}_1$ is a basic graph, contains an induced forbidden graph $\mathbf{B}$ whose vertices are neither all in H_1 nor H_2; without loss of generality H_1 contains at least one loop, its loops form a clique and none of its edges connects two non-loops. It is then easy to verify that $\mathbf{B}$ contains both loops and non-loops. Because the other cases are similar, we prove only that $\mathbf{B}$ is not $\mathbf{B3}$: since vertex d is not adjacent to a it must be in $\mathbf{H}_2$, and similarly for c. Since b is not adjacent to d it must also be in $\mathbf{H}_2$; since non-loops of $\mathbf{H}_1$ are not adjacent to elements of $\mathbf{H}_2$ it follows that a is in $\mathbf{H}_2$ also, a contradiction.

Now we must show that every graph in $\mathcal{L}$ can be obtained from the basic graphs by disjoint union and adjunction of basic graphs. Suppose this is not the case. If $\mathbf{H}$ is a counterexample of minimum size, then obviously it is connected, and it contains at least one loop for otherwise it is a basic irreflexive graph. By Lemma 3.6, $\mathbf{H}$ also contains at least one non-loop.

For $a \in H$ let $N(a)$ denote its set of neighbours. Let $\mathbf{R}(\mathbf{H})$ denote the subgraph of $\mathbf{H}$ induced by its set $R(H)$ of loops, and let $\mathbf{J}(\mathbf{H})$ denote the subgraph induced by $J(H)$, the set of non-loops of $\mathbf{H}$. Since $\mathbf{H}$ is connected and neither $\mathbf{B1}$ nor $\mathbf{B2}$ is an induced subgraph of $\mathbf{H}$, the graph $\mathbf{R}(\mathbf{H})$ is also connected, and furthermore every vertex in $J(H)$ is adjacent to some vertex in $R(H)$. By Lemma 3.6, we know that $\mathbf{R}(\mathbf{H})$ contains at least one universal vertex: let U denote the (non-empty) set of universal vertices of $\mathbf{R}(\mathbf{H})$. Let J denote the set of all $a \in J(H)$ such that $N(a) \cap R(H) \subseteq U$. Let us show that $J \neq \emptyset$. For every $u \in U$, there is $w \in J(H)$ not adjacent to u because otherwise $\mathbf{H}$ is obtained by adjoining u to the rest of $\mathbf{H}$, a contradiction with the choice of $\mathbf{H}$. If this w has a neighbour $r \in R(H) \setminus U$ then there is some $s \in R(H) \setminus U$ not adjacent to r, and the graph induced by $\{w, u, s, r\}$ contains $\mathbf{B2}$ or $\mathbf{B3}$, a contradiction. Hence, $w \in J$. Let $\mathbf{S}$ denote the subgraph of $\mathbf{H}$ induced by $U \cup J$. The graph $\mathbf{S}$ is connected. We claim that the following properties also hold:

(1) if a and b are adjacent non-loops, then $N(a) \cap U = N(b) \cap U$;
(2) if a is in a connected component of the subgraph of $\mathbf{S}$ induced by J with more than one vertex, then for any other $b \in J$, one of $N(a) \cap U, N(b) \cap U$ contains the other.

The first statement holds because $\mathbf{B1}$ is forbidden, and the second follows from the first because $\mathbf{B4}$ is also forbidden. Let $J_1, \ldots, J_k$ denote the different connected components of J in $\mathbf{S}$. By (1) we may let $N(J_i)$ denote the set of common neighbours of members of J_i in U. By (2), we can re-order the J_i's so that for some $1 \le m \le k$ we have $N(J_i) \subseteq N(J_j)$ for all $i \le m$ and all $j > m$, and, in addition, we have $m = 1$ or $|J_i| = 1$ for all $1 \le i \le m$. Let $\mathbf{B}$ denote the subgraph of $\mathbf{S}$ induced by $B = \bigcup_{i=1}^{m} (J_i \cup N(J_i))$, and let $\mathbf{C}$ be the subgraph

of $\mathbf{H}$ induced by $H \setminus B$. We claim that $\mathbf{H} = \mathbf{B} \oslash \mathbf{C}$. For this, it suffices to show that every element in $\bigcup_{i=1}^{m} N(J_i)$ is adjacent to every non-loop $c \in C$. By construction this holds if $c \in J \cap C$. Now suppose this does not hold: then some $x \in J(H) \setminus J$ is not adjacent to some $y \in N(J_i)$ for some $i \leq m$. Since $x \notin J$ we may find some $z \in R(H) \setminus U$ adjacent to x; it is of course also adjacent to y. Since $z \notin U$ there exists some $z' \in R(H) \setminus U$ that is not adjacent to z, but it is of course adjacent to y. If x is adjacent to z', then $\{x, z, z'\}$ induces a subgraph isomorphic to $\mathbf{B2}$, a contradiction. Otherwise, $\{x, z, y, z'\}$ induces a subgraph isomorphic to $\mathbf{B3}$, also a contradiction.

If every J_i with $i \leq m$ contains a single element, notice that $\mathbf{B}$ is a basic graph: indeed, removing all edges between its loops yields a bipartite irreflexive graph which contains neither the path of length 5 nor the 6-cycle, since $\mathbf{B}$ contains neither $\mathbf{B5}$ nor $\mathbf{B6}$. Since this contradicts our hypothesis on $\mathbf{H}$, we conclude that $m = 1$. But this means that $N(J_1)$ is a set of universal vertices in $\mathbf{H}$. Let u be such a vertex and let D denote its complement in $\mathbf{H}$: clearly $\mathbf{H}$ is obtained as the adjunction of the single loop u to D, contradicting our hypothesis. This concludes the proof. ∎

4. Classification results

Recall the standard numbering of types: (1) *unary*, (2) *affine* , (3) *Boolean*, (4) *lattice* and (5) *semilattice*. We will need the following auxiliary result (which is well known). Note that the assumptions of this lemma effectively say that CSP($\mathbf{T}$) can simulate the graph k-colouring problem (with $k = |U|$) or the directed *st*-connectivity problem.

Lemma 4.1. *Let* $\mathbf{S}, \mathbf{T}$ *be structures, let* $s_1, s_2 \in S$, *and let* $R = \{(f(s_1), f(s_2)) \mid f : \mathbf{S} \to \mathbf{T}\}$.
  (1) *If* $R = \{(x, y) \in U^2 \mid x \neq y\}$ *for some subset* $U \subseteq T$ *with* $|U| \geq 3$ *then* $\mathcal{V}(\mathbb{A}_{\mathbf{T}})$ *admits type 1.*
  (2) *If* $R = \{(t, t), (t, t'), (t', t')\}$ *for some distinct* $t, t' \in T$ *then* $\mathcal{V}(\mathbb{A}_{\mathbf{T}})$ *admits at least one of the types 1, 4, 5.*

Proof [sketch]: The assumption of this lemma implies that $\mathbb{A}_{\mathbf{T}}$ has a subalgebra (induced by U and $\{t, t'\}$, respectively) such that all operations of the subalgebra preserve the relation R. It is well-known (see, e.g., [17]) that all operations preserving the disequality relation on U are essentially unary, while it is easy to check that the order relation on a 2-element set cannot admit operations satisfying identities (2.1)–(2.3), so one can use Lemma 2.1. ∎

The following lemma connects the characterisation of bi-arc graphs given in [4] with a type-omitting condition.

Lemma 4.2. *Let* $\mathbf{H}$ *be a graph. Then the following conditions are equivalent:*
  (1) *the variety* $\mathcal{V}(\mathbb{H})$ *omits type 1;*
  (2) *the graph* $\mathbf{H}$ *admits a conservative majority operation;*
  (3) *the graph* $\mathbf{H}$ *is a bi-arc graph.*

The results summarised in the following theorem are known (or easily follow from known results, with a little help from Lemma 4.2).

Theorem 4.3. *Let* $\mathbf{H}$ *be a graph.*
  • *If* $typ(\mathcal{V}(\mathbb{H}))$ *admits type 1, then* $\neg \mathrm{CSP}(\mathbf{H}^L)$ *is not expressible in Datalog and* $\mathrm{CSP}(\mathbf{H}^L)$ *is* NP-*complete (under first-order reductions);*

- *if $typ(\mathcal{V}(\mathbb{H}))$ omits type 1 but admits type 4 then $\neg\,\mathrm{CSP}(\mathbf{H}^L)$ is not expressible in symmetric Datalog but is expressible in linear Datalog, and $\mathrm{CSP}(\mathbf{H}^L)$ is NL-complete (under first-order reductions.)*

Proof. The first statement is shown in [22]. If the variety omits type 1, then $\mathbf{H}^L$ admits a majority operation by Lemma 4.2 and then $\neg\,\mathrm{CSP}(\mathbf{H}^L)$ is expressible in linear Datalog by [10]; in particular the problem is in NL. If, furthermore, the variety admits type 4, then $\neg\,\mathrm{CSP}(\mathbf{H}^L)$ is not expressible in symmetric Datalog and is NL-hard by results in [22]. ∎

By Lemma 2.1, the presence of a majority operation in $\mathbb{H}$ implies that $typ(\mathcal{V}(\mathbb{H}))$ can contain only types 3 and 4. Type 4 is dealt with in Theorem 4.3, so it remains to investigate graphs $\mathbf{H}$ with $typ(\mathcal{V}(\mathbb{H})) = \{3\}$.

The next theorem is the main result of this paper.

Theorem 4.4. *Let $\mathbf{H}$ be a graph. Then the following conditions are equivalent:*
 (1) $\mathbf{H}$ *admits conservative operations satisfying (2.1)–(2.3) for $n = 3$;*
 (2) $\mathbf{H}$ *admits conservative operations satisfying (2.1)–(2.3) for some $n \geq 1$;*
 (3) $typ(\mathcal{V}(\mathbb{H})) = \{3\}$;
 (4) $\mathbf{H} \in \mathcal{L}$;
 (5) $\neg\,\mathrm{CSP}(\mathbf{H}^L)$ *is definable in symmetric Datalog.*
If the above holds then $\mathrm{CSP}(\mathbf{H}^L)$ is in the complexity class L.

Proof [sketch]: (1) $\Rightarrow$ (2) is trivial. If (2) holds then by Lemma 2.1 $\mathcal{V}(\mathbb{H})$ omits types 1, 4, and 5. By Lemma 4.2, $\mathbf{H}$ admits a majority operation, so Lemma 2.1 implies that $\mathcal{V}(\mathbb{H})$ also omits type 2; hence (3) holds. Implication (3)$\Rightarrow$(4) is the content of Lemma 4.5 below, and (5) implies (3) by a result of [22]. By using Theorem 3.7, one can show that (4) implies both (1) and (5). Finally, definability in symmetric Datalog implies membership in L by [12]. ∎

Lemma 4.5. *If $\mathbf{H} \notin \mathcal{L}$ then $typ(\mathcal{V}(\mathbb{H})) \neq \{3\}$.*

Proof. By Theorem 9.15 of [19], $typ(\mathcal{V}(\mathbb{H})) = \{3\}$ if and only if $\mathbf{H}$ admits a sequence of conservative operations satisfying certain identities (in the spirit of (2.1)–(2.3)). By conservativity, such operations can be restricted to any subset of H while satisfying the same identities, so the property $typ(\mathcal{V}(\mathbb{H})) = \{3\}$ is inherited by induced subgraphs. It follows that it is enough to prove this lemma for the forbidden graphs from Definition 3.3.

For the irreflexive odd cycles, the lemma follows immediately from the main results of [3, 23]. The proof of Theorem 3.1 of [13] shows that the conditions of Lemma 4.1(1) are satisfied by (some $\mathbf{S}, s_1, s_2$ and) $\mathbf{T} = \mathbf{F}^L$ where $\mathbf{F}$ is the irreflexive 6-cycle. One can check that the reflexive 4-cycle is not a bi-arc graph, so we can apply Lemma 4.2 in this case.

For the remaining forbidden graphs $\mathbf{F}$ from Definition 3.3, we use Lemma 4.1(2) with $\mathbf{T} = \mathbf{F}^L$. In each case, the binary relation of the structure $\mathbf{S}$ will be a short undirected path, and s_1, s_2 will be the endpoints of the path. We will represent such a structure $\mathbf{S}$ by a sequence of subsets of F (indicating lists assigned to vertices of the path). It can be easily checked that, in each case, the relation R defined as in Lemma 4.1 is of the required form.

If $\mathbf{F}$ is the reflexive path of length 3, say $a - b - c - d$, then $\mathbf{S} = ac - bc - ad - ac$. If $\mathbf{F}$ is the irreflexive path of length 5, say $a - b - c - d - e - f$ then $\mathbf{S} = ae - bd - ce - bf - ae$. For graphs $\mathbf{B1} - \mathbf{B6}$, we use notation from Fig. 1. For $\mathbf{B1}$, $\mathbf{S} = bc - bc - ab - ab - bc$. For $\mathbf{B2}$, $\mathbf{S} = bc - ac - ab - bc$. For $\mathbf{B3}$, $\mathbf{S} = bc - ad - bd - bc$. For $\mathbf{B4}$, $\mathbf{S} = ae - bd - cd - ae$. Finally, for both $\mathbf{B5}$ and $\mathbf{B6}$, $\mathbf{S} = ac - b'c' - ab - a'c' - ac$. ∎

For completeness' sake, we describe graphs whose list homomorphism problem is definable in first-order logic (equivalently, is in AC^0, see [6].) By results in [22], any problem $CSP(\mathbf{T})$ is either first-order definable or L-hard under FO reductions. Hence, it follows from Theorem 4.4 that, for a graph $\mathbf{H} \in \mathcal{L}$, the list homomorphism problem for $\mathbf{H}$ is either first-order definable or L-complete.

We need the following characterisation of structures whose CSP is first-order definable [21]. Let $\mathbf{T}$ be a relational structure and let $a, b \in T$. We say that b *dominates* a in $\mathbf{T}$ if for any relation R of $\mathbf{T}$, and any tuple $\bar{t} \in R$, replacement of any occurrence of a by b in $\bar{t}$ will yield a tuple of R. Recall the definition of a direct power of a structure from Subsection 2.1. If $\mathbf{T}$ is a relational structure, we say that the structure $\mathbf{T}^2$ *dismantles to the diagonal* if there exists a sequence of elements $\{a_0, \ldots, a_n\} = T^2 \setminus \{(a, a) : a \in T\}$ such that, for all $0 \leq i \leq n$, a_i is dominated in $\mathbf{T}_i$, where $\mathbf{T}_0 = \mathbf{T}^2$ and $\mathbf{T}_i$ is the substructure of $\mathbf{T}^2$ induced by $T^2 \setminus \{a_0, \ldots, a_{i-1}\}$ for $i > 0$.

Lemma 4.6 ([21]). *Let $\mathbf{T}$ be a core relational structure. Then $CSP(\mathbf{T})$ is first-order definable if and only if $\mathbf{T}^2$ dismantles to the diagonal.*

Theorem 4.7. *Let $\mathbf{H}$ be a graph. Then $CSP(\mathbf{H}^L)$ is first-order definable if and only if $\mathbf{H}$ has the following form: H is the disjoint union of two sets L and N such that (i) L is the set of loops of $\mathbf{H}$ and induces a complete graph, (ii) N is the set of non-loops of $\mathbf{H}$ and induces a graph with no edges, and (iii) $N = \{x_1, \ldots, x_m\}$ can be ordered so that the neighbourhood of x_i is contained in the neighbourhood of x_{i+1} for all $1 \leq i \leq m-1$.*

Proof. We first prove that conditions (i) and (ii) are necessary. Notice that if $CSP(\mathbf{H}^L)$ is first-order definable then so is $CSP(\mathbf{K}^L)$ for any induced substructure $\mathbf{K}$ of $\mathbf{H}$. Let x and y be distinct vertices of $\mathbf{H}$ and let $\mathbf{K}^L$ be the substructure of $\mathbf{H}^L$ induced by $\{x, y\}$. If x and y are non-adjacent loops, then $\theta(\mathbf{K}) = \{(x, x), (y, y)\}$ the equality relation on $\{x, y\}$; if x and y are adjacent non-loops, then $\theta(\mathbf{K}) = \{(x, y), (y, x)\}$, the adjacency relation of the complete graph on 2 vertices. It is well known (and can be easily derived from Lemma 4.6) that neither of these classes $CSP(\mathbf{K}^L)$ is first-order definable. It follows that the loops of $\mathbf{H}$ induce a complete graph and the non-loops induce a graph with no edges.

Now we prove (iii) is necessary. Suppose for a contradiction that there exist distinct elements x and y of N and elements n and m of L such that m is adjacent to x but not to y, and n is adjacent to y but not to x. Then $CSP(\mathbf{G})$ is first-order definable, where $\mathbf{G}$ is the substructure of $\mathbf{H}^L$ induced by $\{x, y, m, n\}$. By Lemma 4.6, $\mathbf{G}^2$ dismantles to the diagonal. Then (x, y) must be dominated by one of (x, x), (y, x) or (y, y), since domination respects the unary relation $\{x, y\}^2$ (on G^2). But (m, n) is a neighbour of (x, y) and none of the other three, a contradiction.

For the converse: we show that we can dismantle $(\mathbf{H}^L)^2$ to the diagonal. Let $x \in H$: then (x_1, x) and (x, x_1) are dominated by (x, x). Suppose that we have dismantled every element containing a coordinate equal to x_i with $i \leq j-1$: if x is any element of H such that the elements (x_j, x) and (x, x_j) remain, then either x is a loop or $x = x_k$ with $k \geq j$; in any case the elements (x_j, x_k) and (x_k, x_j) are dominated by (x, x). In this way we can remove all pairs (x, y) with one of x or y a non-loop. For the remaining pairs, notice that if u and v are any loops then (u, v) is dominated (in what remains of $(\mathbf{H}^L)^2$) by (u, u). ∎

Finally, given a graph $\mathbf{H}$, it can be decided in polynomial time which of the different cases delineated in Theorems 4.3, 4.4, 4.7 the list homomorphism problem for $\mathbf{H}$ satisfies. Indeed, it is known that bi-arc graphs can be recognised in polynomial time (see [14]). Assume that $\mathbf{H}$ is a bi-arc graph: the forbidden substructure definition of the class $\mathcal{L}$ gives

an AC^0 algorithm to recognise them; and those graphs whose list homomorphism problem is first-order definable can be recognised in polynomial time by results of [21].

References

[1] E. Allender, M. Bauland, N. Immerman, H. Schnoor, and H. Vollmer. The complexity of satisfiability problems: Refining Schaefer's theorem. *Journal of Computer and System Sciences*, 75(4):245–254, 2009.

[2] L. Barto and M. Kozik. Constraint satisfaction probllems of bounded width. In *FOCS'09*, 2009.

[3] L. Barto, M. Kozik, and T. Niven. The CSP dichotomy holds for digraphs with no sources and no sinks (A positive answer to a conjecture of Bang-Jensen and Hell). *SIAM J. Comput.*, 38(5):1782–1802, 2009.

[4] R. Brewster, T. Feder, P. Hell, J. Huang, and G. MacGillavray. Near-unanimity functions and varieties of reflexive graphs. *SIAM J. Discrete Math.*, 22:938–960, 2008.

[5] A. Bulatov. Tractable conservative constraint satisfaction problems. In *LICS'03*, pages 321–330, 2003.

[6] A. Bulatov, A. Krokhin, and B. Larose. Dualities for constraint satisfaction problems. In *Complexity of Constraints*, volume 5250 of *LNCS*, pages 93–124. 2008.

[7] A. Bulatov and M. Valeriote. Recent results on the algebraic approach to the CSP. In *Complexity of Constraints*, volume 5250 of *LNCS*, pages 68–92. 2008.

[8] D. Cohen and P. Jeavons. The complexity of constraint languages. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 8. Elsevier, 2006.

[9] V. Dalmau. Linear Datalog and bounded path duality for relational structures. *Logical Methods in Computer Science*, 1(1), 2005. (electronic).

[10] V. Dalmau and A. Krokhin. Majority constraints have bounded pathwidth duality. *European Journal of Combinatorics*, 29(4):821–837, 2008.

[11] V. Dalmau and B. Larose. Maltsev + Datalog ⇒ Symmetric Datalog. In *LICS'08*, pages 297–306, 2008.

[12] L. Egri, B. Larose, and P. Tesson. Symmetric Datalog and constraint satisfaction problems in Logspace. In *LICS'07*, pages 193–202, 2007.

[13] T. Feder, P. Hell, and J. Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19:487–505, 1999.

[14] T. Feder, P. Hell, and J. Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42:61–80, 2003.

[15] T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM J. Comput.*, 28:57–104, 1998.

[16] F. Gurski. Characterizations of co-graphs defined by restricted NLC-width or clique-width operations. *Discrete Mathematics*, 306(2):271–277, 2006.

[17] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[18] P. Hell and J. Nešetřil. Colouring, constraint satisfaction, and complexity. *Computer Science Review*, 2(3):143–163, 2008.

[19] D. Hobby and R.N. McKenzie. *The Structure of Finite Algebras*. AMS, Providence, R.I., 1988.

[20] Ph.G. Kolaitis and M.Y. Vardi. A logical approach to constraint satisfaction. In *Complexity of Constraints*, volume 5250 of *LNCS*, pages 125–155. 2008.

[21] B. Larose, C. Loten, and C. Tardif. A characterisation of first-order constraint satisfaction problems. *Logical Methods in Computer Science*, 3(4), 2007. (electronic).

[22] B. Larose and P. Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theoretical Computer Science*, 410(18):1629–1647, 2009.

[23] M. Maróti and R. McKenzie. Existence theorems for weakly symmetric operations. *Algebra Univ.*, 59(3-4):463–489, 2008.

[24] T.J. Schaefer. The complexity of satisfiability problems. In *STOC'78*, pages 216–226, 1978.

[25] M. Valeriote. A subalgebra intersection property for congruence-distributive varieties. *Canadian Journal of Mathematics*, 61(2):451–464, 2009.

[26] László Zádori and Benoit Larose. Bounded width problems and algebras. *Algebra Univ.*, 56(3-4):439–466, 2007.